MyFirebaseChat Android App

Overview

My Firebase Chat app is purely developed using Native Android Language. This also known as real chatting app between two people using Firebase Database. App having premium features that enhance user experience more clearly. User can send sms, able to see Online/Offline status, typing indicator, support in-build emoji's, display date/time wise messages.

This app uses Firebase as backend part, there were no use of any third party admin panel or backend(PHP, Java, or any other framework). Source code is very easy to understand, customize and re-skin the app for their personal use.

Installation

Required Software:

- Android Studio
- Java
- Google Firebase

Project Setup

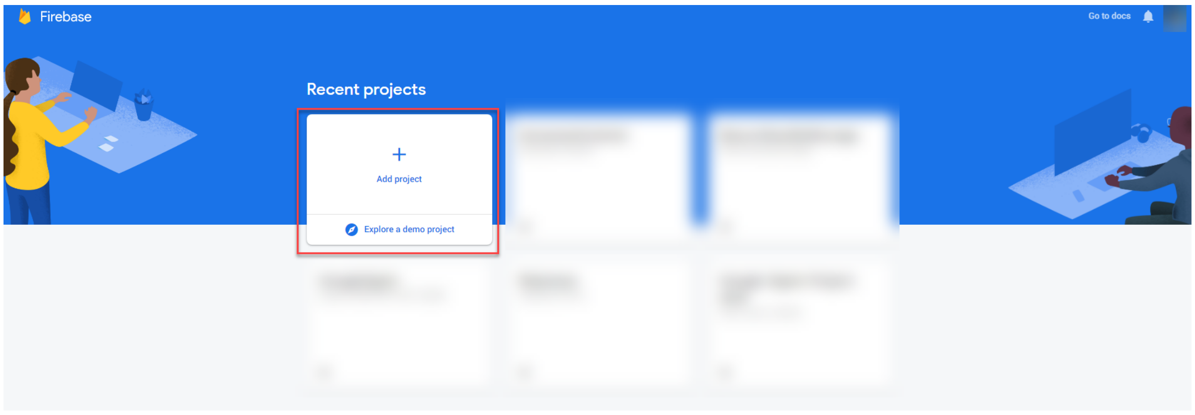
Let we setup the project and we will go through step by step.

Latest Android Studio version is recommended, which can be downloaded from here: https://developer.android.com/studio?hl=es

To get google-service.json, first we need to create new project setup from Firebase console.

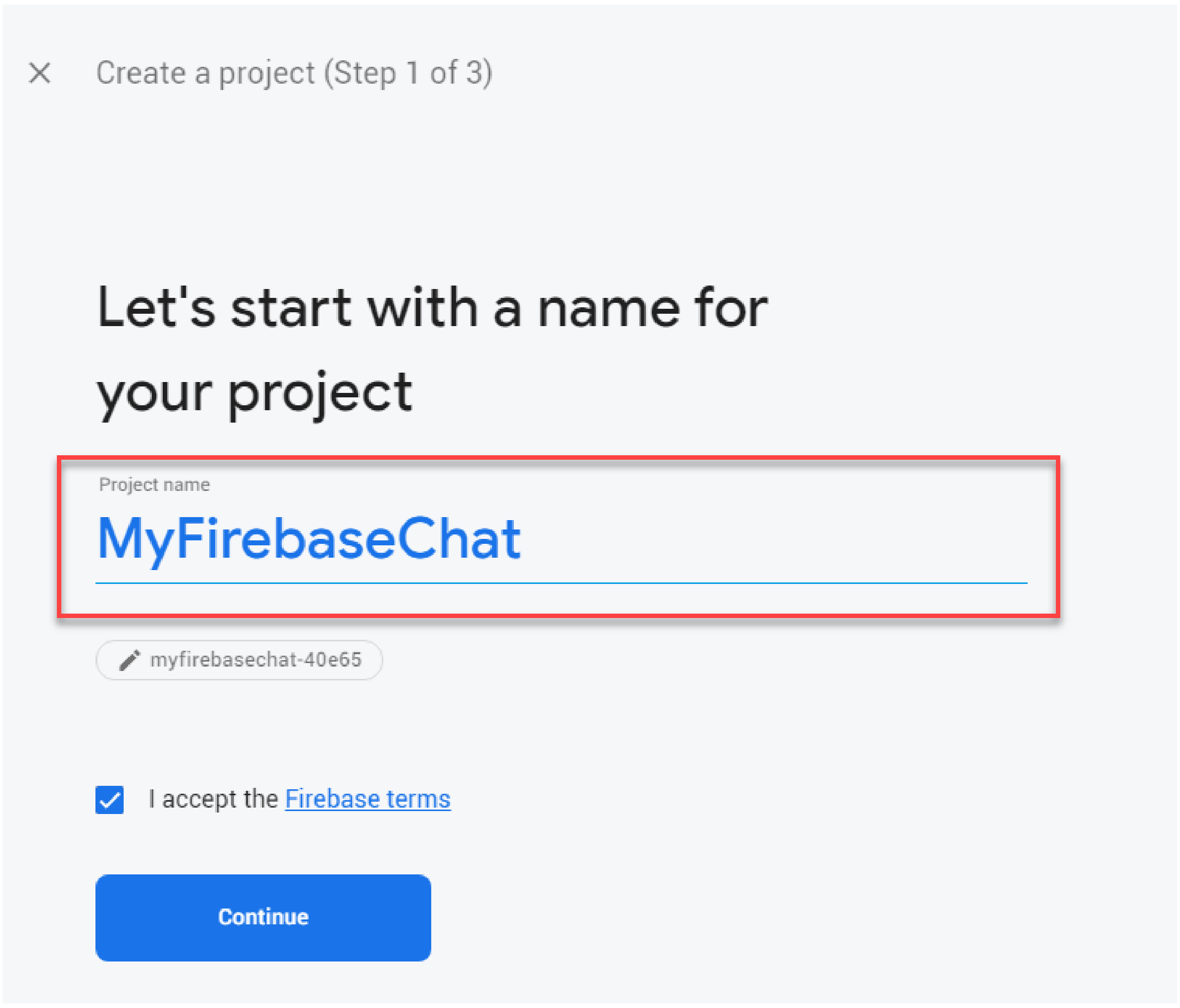
Step 1: Firebase Setup

Open Firebase console and login with your gmail account : https://console.firebase.google.com/u and than click on 'Add Project' option.



Step 2: Project Name

Enter your project name and Continue


× Create a project (Step 1 of 3)

Let's start with a name for your project

Project name
MyFirebaseChat

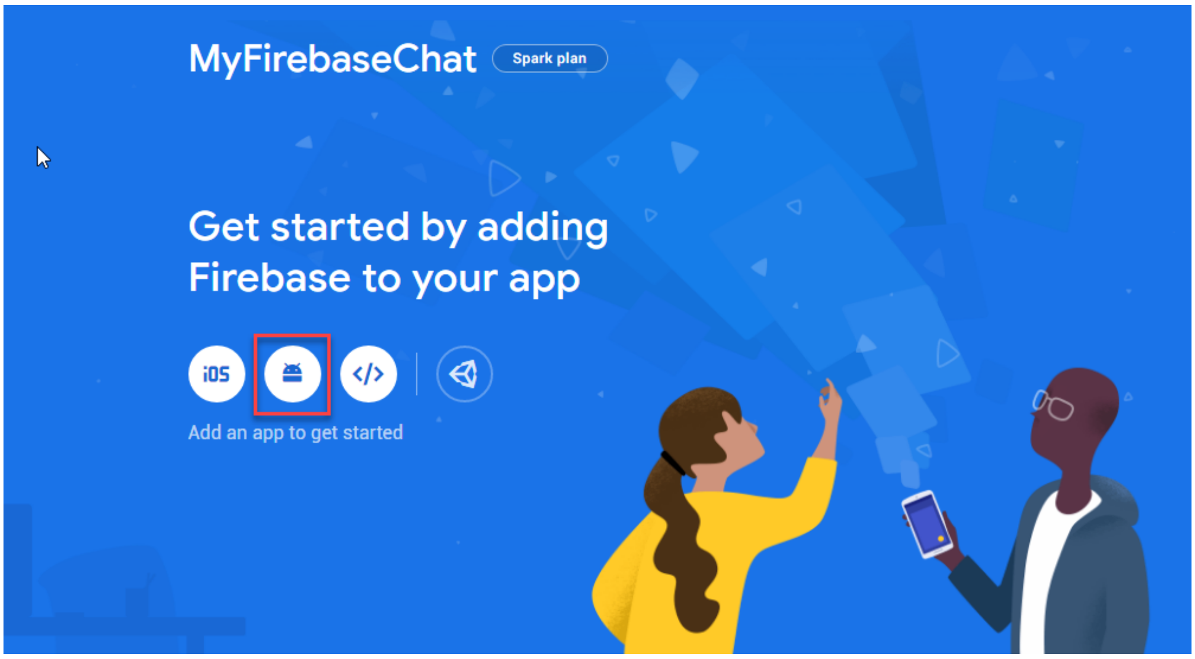
myfirebasechat-40e65

☑ I accept the Firebase terms

Continue

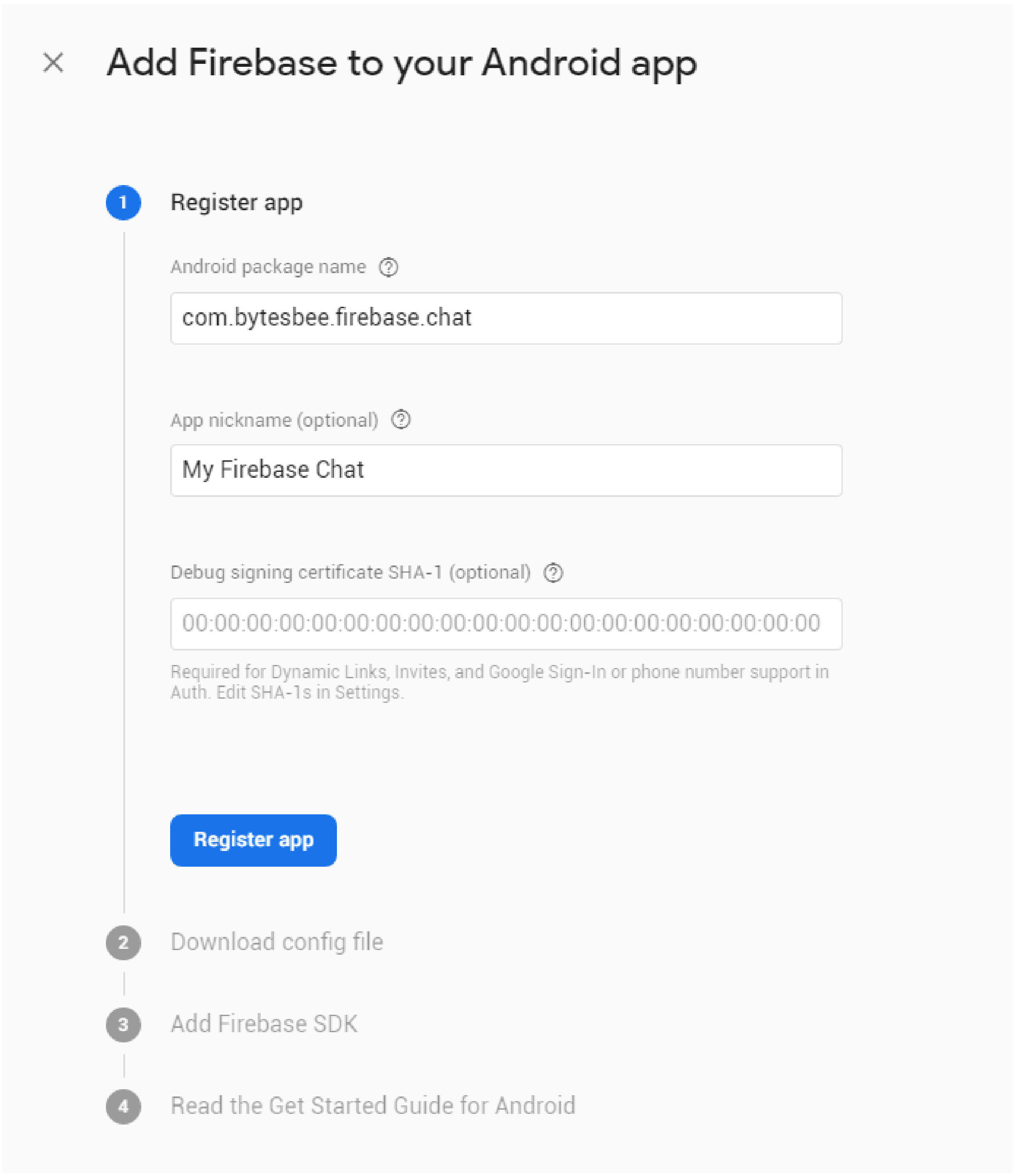
Step 3: Adding app

Select Android icon to add our project details.


MyFirebaseChat
Spark plan
Get started by adding Firebase to your app
iOS
</>
Add an app to get started

Step 4: Register app

Add package name, nickname(optional), SHA-1(optional) and click on Register app



Add Firebase to your Android app

1 Register app

Android package name

com.bytesbee.firebase.chat

App nickname (optional)

My Firebase Chat

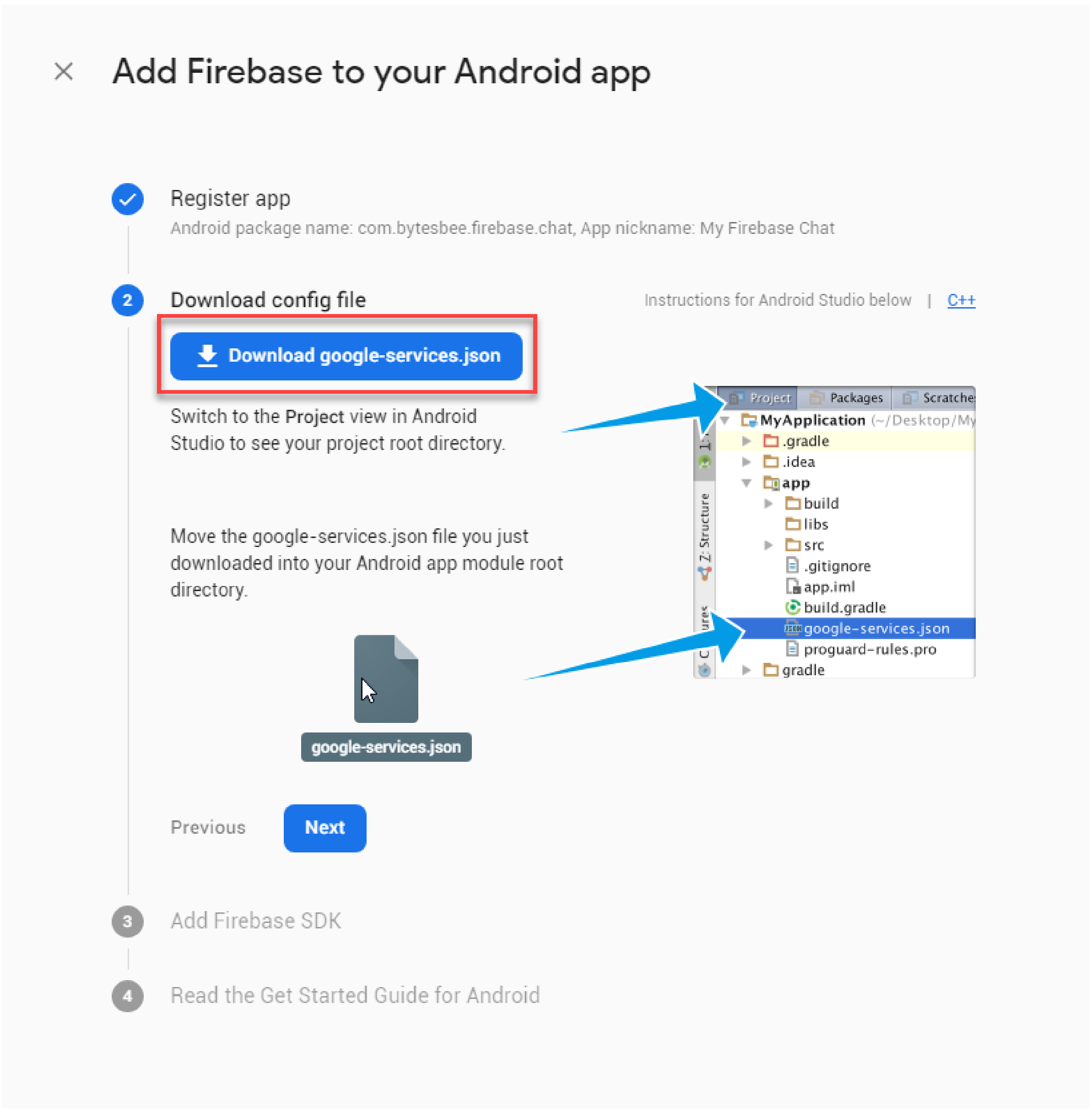
Debug signing certificate SHA-1 (optional)

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00

Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Step 5: Download google-service.json

Download **google-service.json** file and place inside your project


Add Firebase to your Android app

Register app

Android package name: com.bytesbee.firebase.chat, App nickname: My Firebase Chat

Download config file

Instructions for Android Studio below | C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

Previous

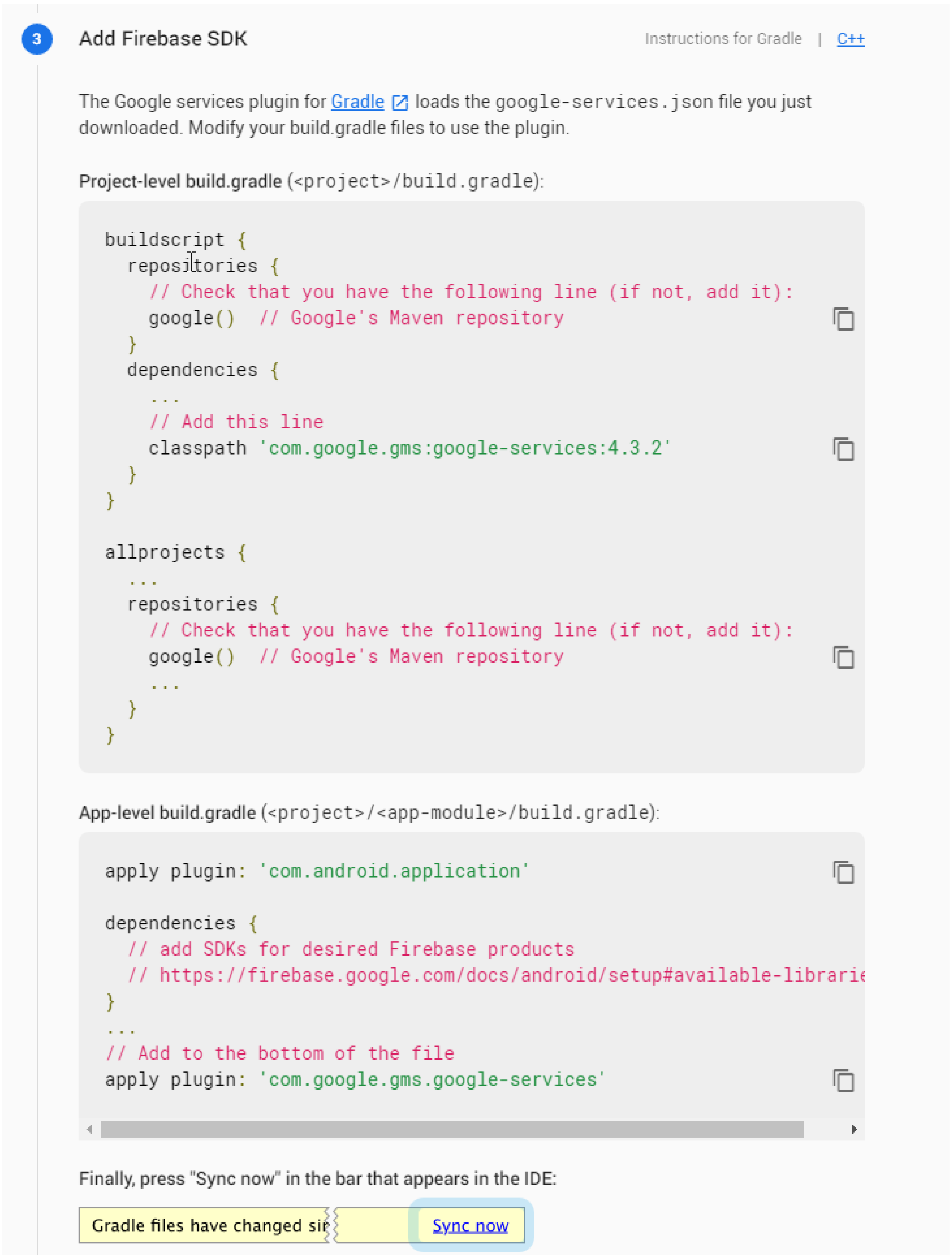
Next

3 Add Firebase SDK

4 Read the Get Started Guide for Android

Step 6: Add Firebase SDK

Add Firebase SDK required plugin inside project-level and app-level build.gradle files.



3 Add Firebase SDK

Instructions for Gradle | C++

The Google services plugin for Gradle loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.2'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

App-level build.gradle (`<project>/<app-module>/build.gradle`):

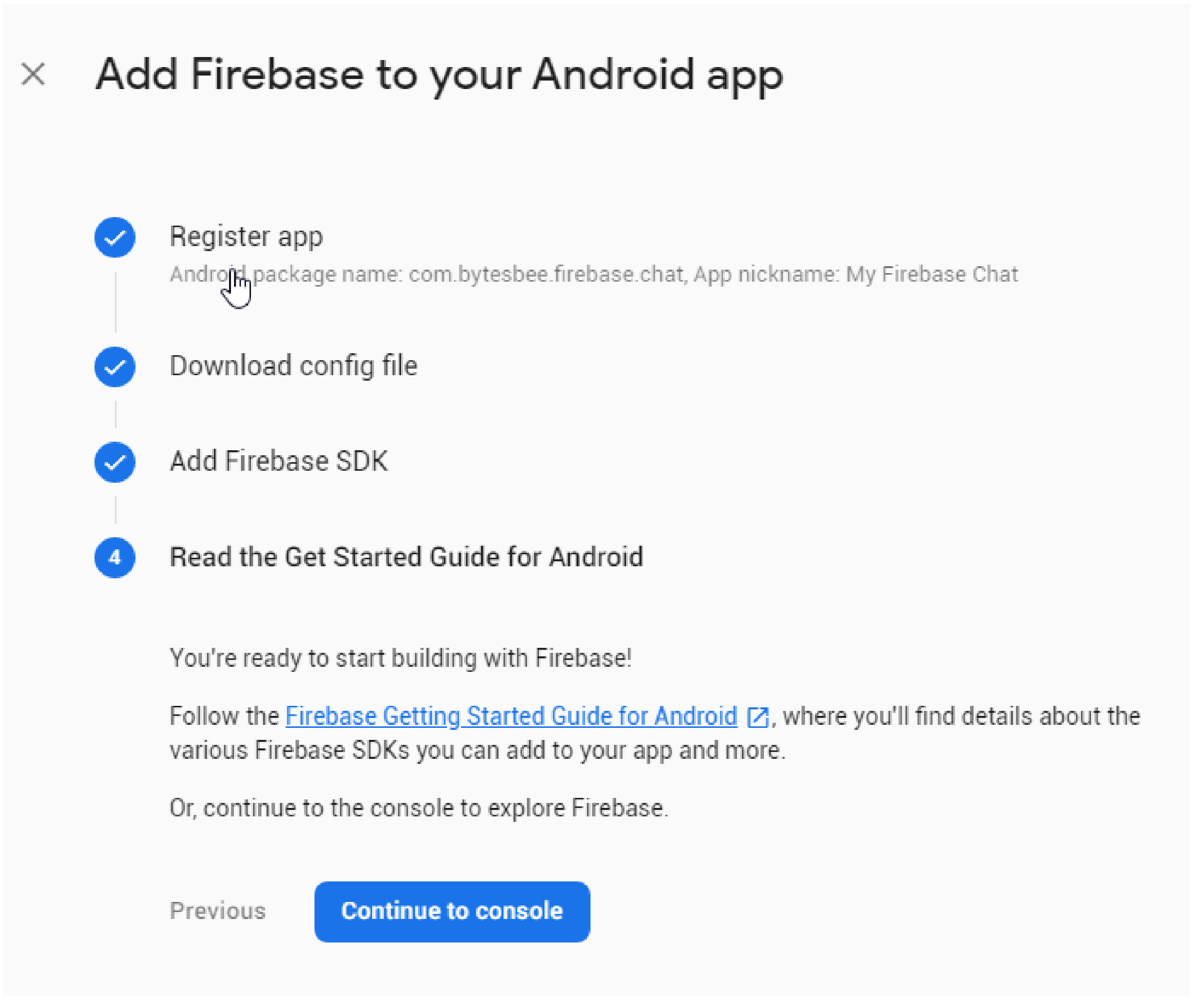
```
apply plugin: 'com.android.application'

dependencies {
  // add SDKs for desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-librarie
}
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Finally, press "Sync now" in the bar that appears in the IDE:

Step 7: Continue to console

Click on Continue to console for further steps.

Now we are setup for Firebase Login, Database, Storage and other required things which is used in our project.



Add Firebase to your Android app

Register app

Android package name: com.bytesbee.firebase.chat, App nickname: My Firebase Chat

Download config file

Add Firebase SDK

4 Read the Get Started Guide for Android

You're ready to start building with Firebase!

Follow the Firebase Getting Started Guide for Android, where you'll find details about the various Firebase SDKs you can add to your app and more.

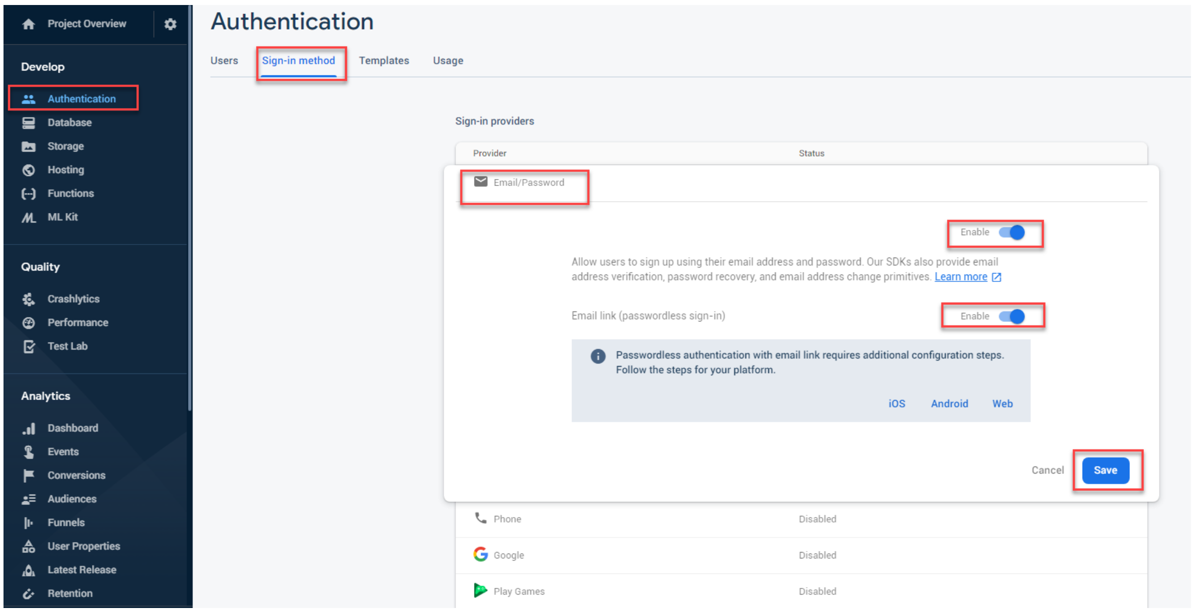
Or, continue to the console to explore Firebase.

Step 8: Authentication

We are provide two provider to enable it. i.e. Email and Google sign in method

A) Email/Password

Add Sign in method as Email



Authentication

Sign-in providers

Provider	Status
Email/Password	Enable
Phone	Disabled
Google	Disabled
Play Games	Disabled

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. Learn more

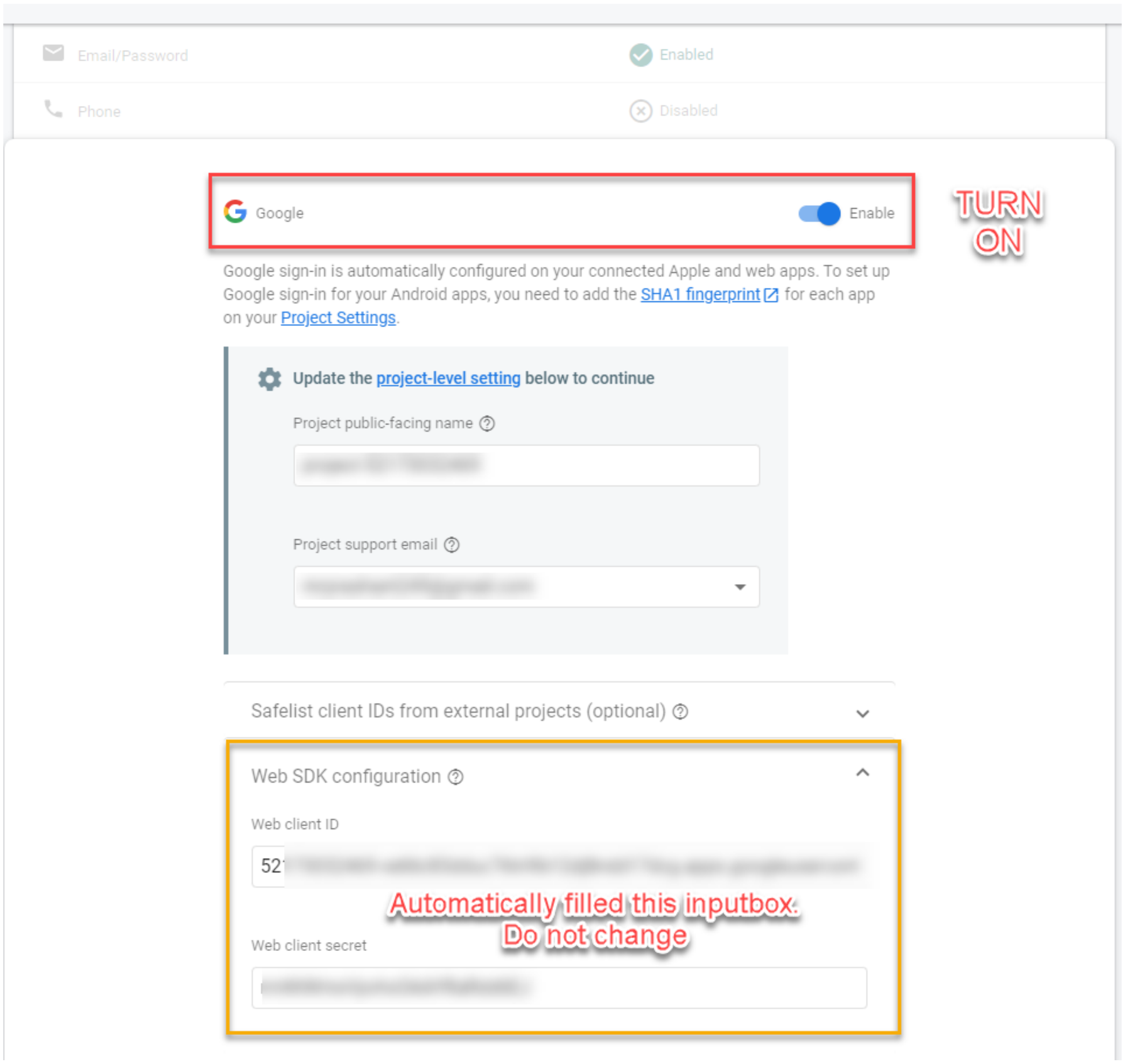
Email link (passwordless sign-in) Enable

Passwordless authentication with email link requires additional configuration steps. Follow the steps for your platform.

iOS Android Web

B) Google Sign In

Enable the Google option, and automatically filled the Web SDK configuration portion. Do not change there. And click on SAVE button.


Email/Password — Enabled
Phone — Disabled
Google — Enable

Google sign-in is automatically configured on your connected Apple and web apps. To set up Google sign-in for your Android apps, you need to add the SHA1 fingerprint for each app on your Project Settings.

Update the project-level setting below to continue

Project public-facing name

Project support email

Safelist client IDs from external projects (optional)

Web SDK configuration

Web client ID

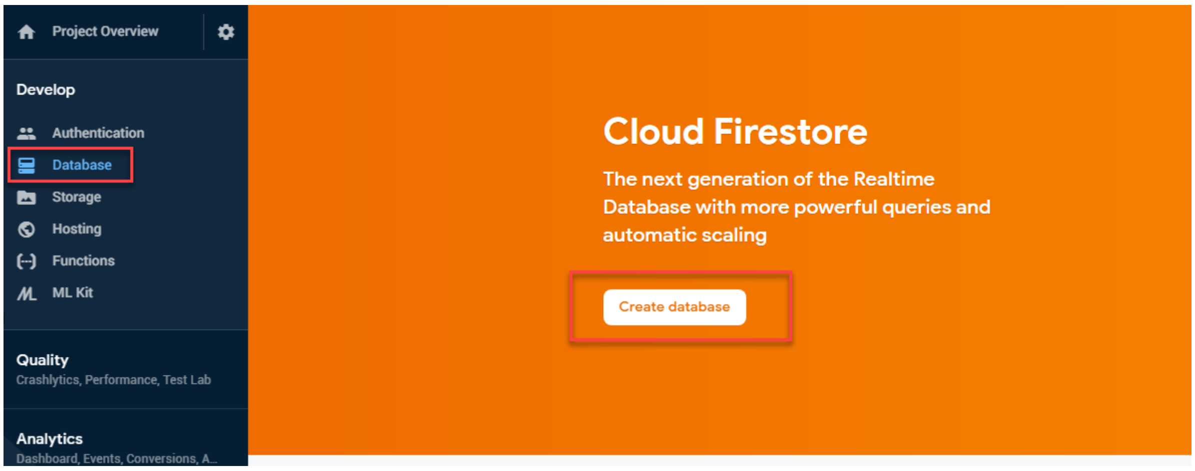
52

Web client secret

For more information, Follow this link : https://firebase.google.com/docs/auth/android/google-signin

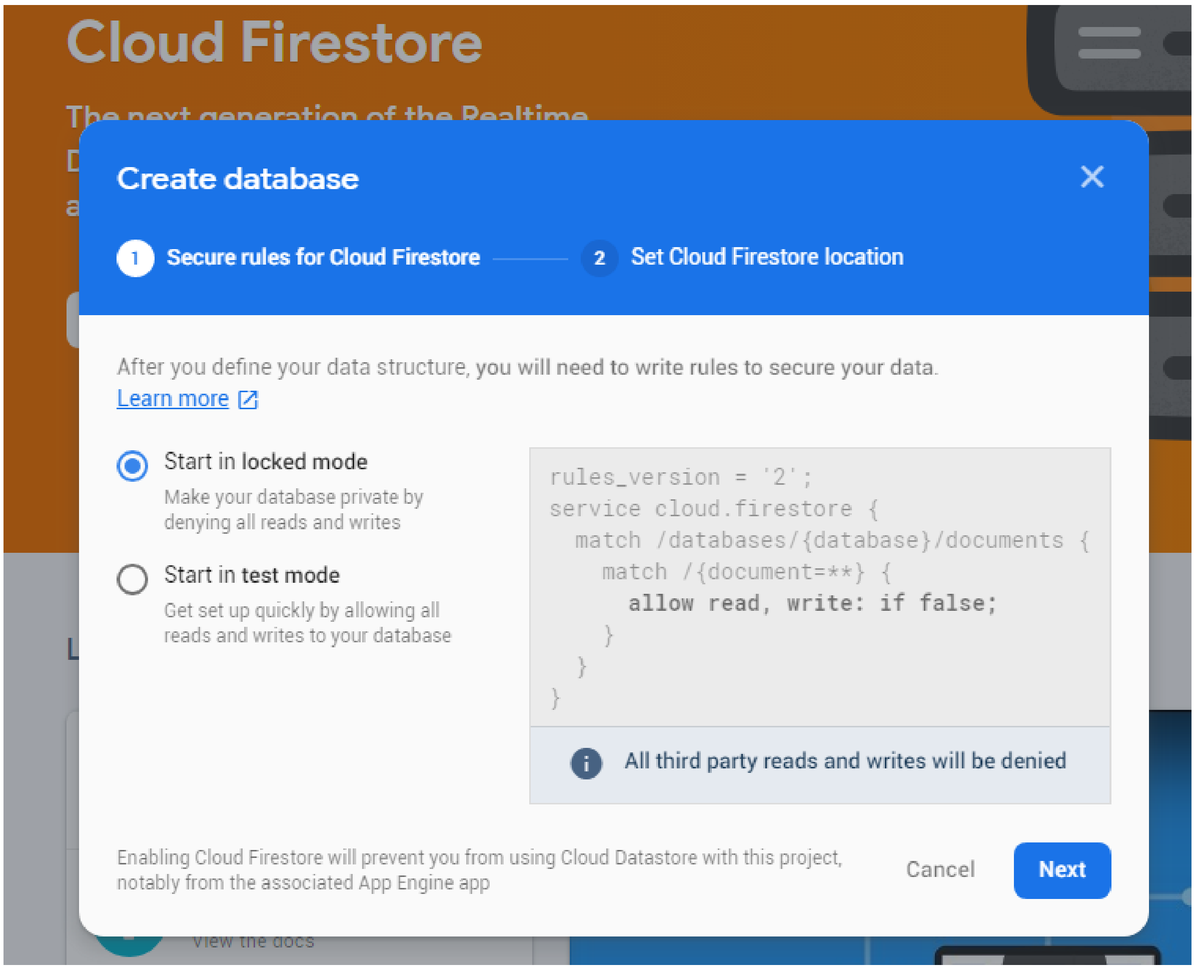
Step 9: Create Database

Create new Database same as shown in screenshot for storing our user's and chat details


Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab
Analytics
Dashboard, Events, Conversions, A...
Cloud Firestore
The next generation of the Realtime Database with more powerful queries and automatic scaling
Create database

Step 10: Select Mode

Click on Next


Cloud Firestore
The next generation of the Realtime
Create database
1 Secure rules for Cloud Firestore
2 Set Cloud Firestore location

After you define your data structure, you will need to write rules to secure your data.
Learn more

[✓ Start in locked mode]
Make your database private by denying all reads and writes

Start in test mode
Get set up quickly by allowing all reads and writes to your database

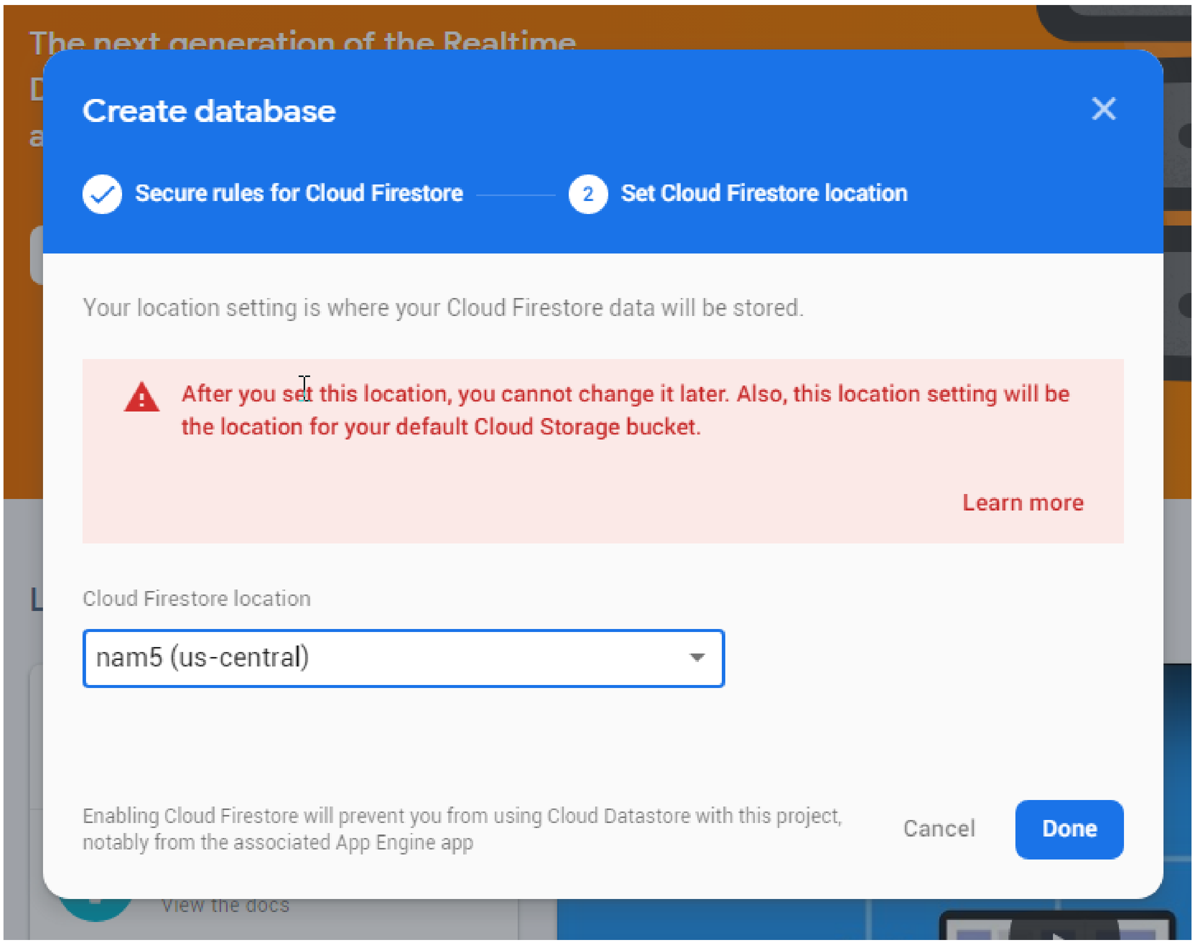
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

All third party reads and writes will be denied

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Step 11: Location

Click on Done



Create database
Secure rules for Cloud Firestore
2 Set Cloud Firestore location

Your location setting is where your Cloud Firestore data will be stored.

After you set this location, you cannot change it later. Also, this location setting will be the location for your default Cloud Storage bucket.

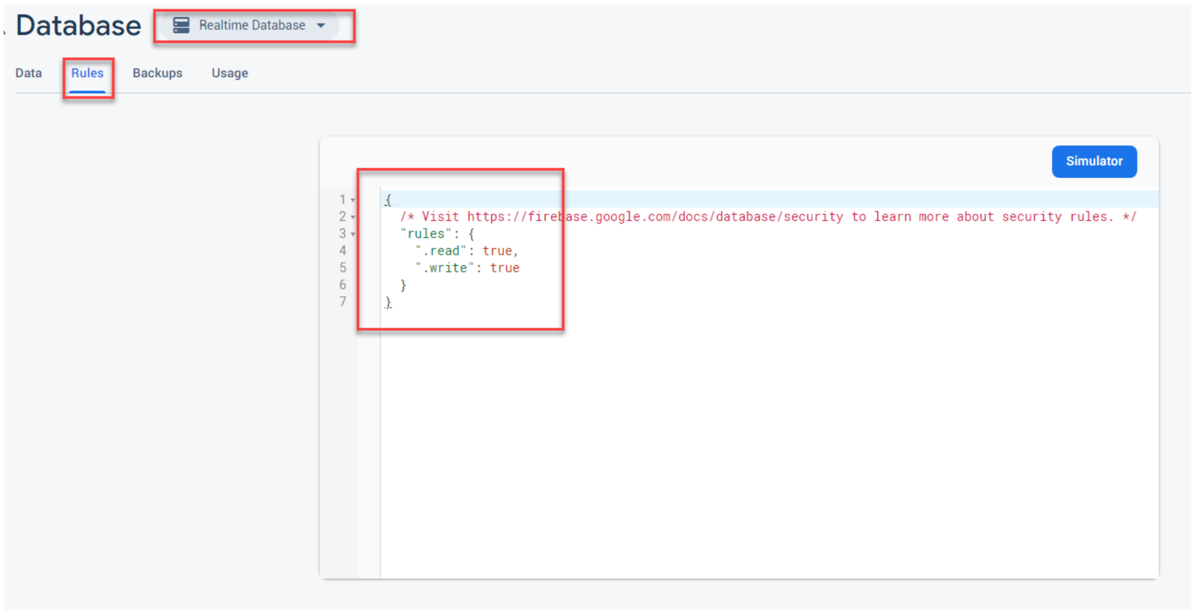
Cloud Firestore location: nam5 (us-central)

Enabling Cloud Firestore will prevent you from using Cloud Datastore with this project, notably from the associated App Engine app

Step 12: Database Rule

Change mode to Realtime Database, and click on Rules tab to Set below Database Rules

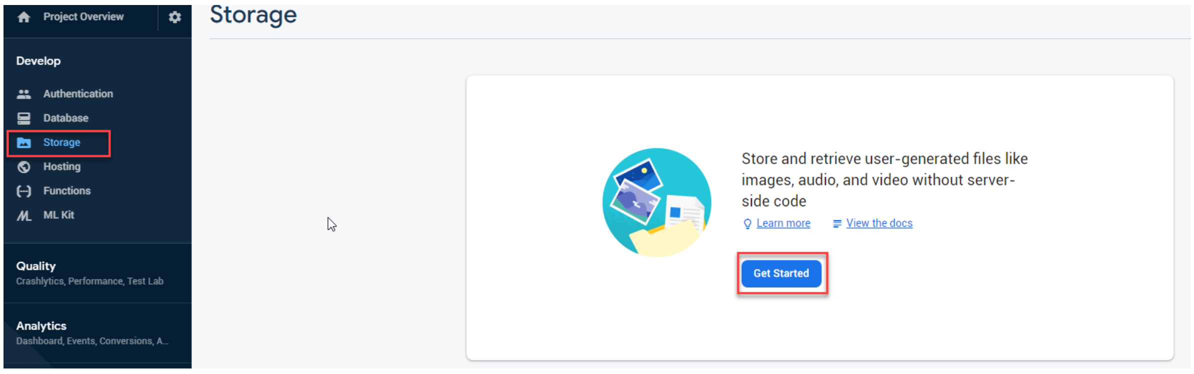
```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about
security rules. */
  "rules": {
     ".read": true,
     ".write": true
  }
}
```



```
{
  /* Visit https://firebase.google.com/docs/database/security to learn more about security rules. */
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Step 13: Enable Storage

Create Storage for storing images and other files based on requirement

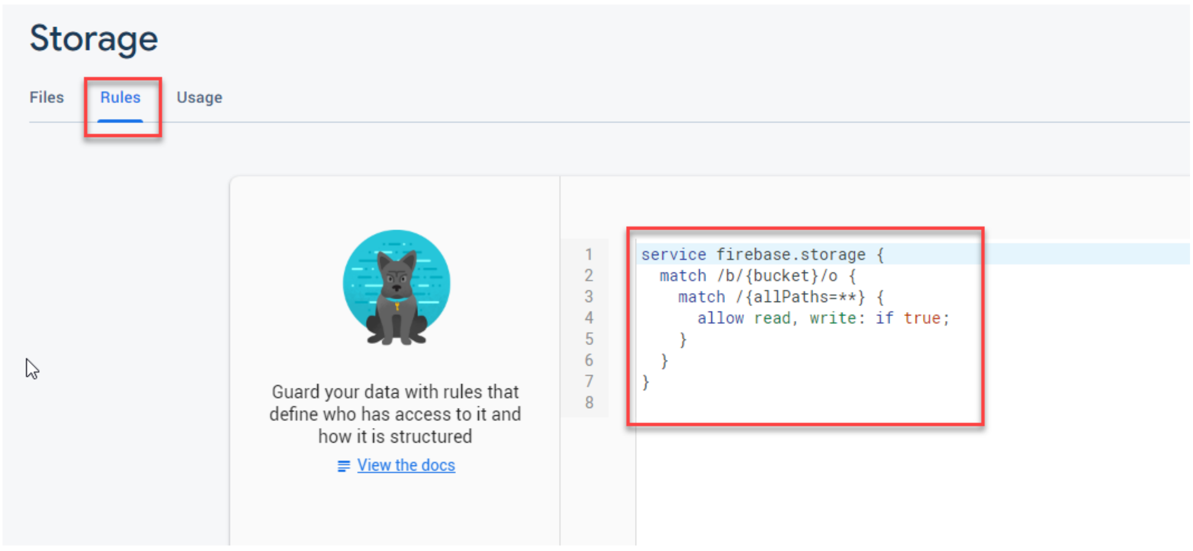


Store and retrieve user-generated files like images, audio, and video without server-side code

Step 14: Storage rules

Make Rules same as given screenshot, mentioned here.

```
service firebase.storage {
   match /b/{bucket}/o {
      match /{allPaths=**} {
        allow read, write: if true;
      }
   }
}
```

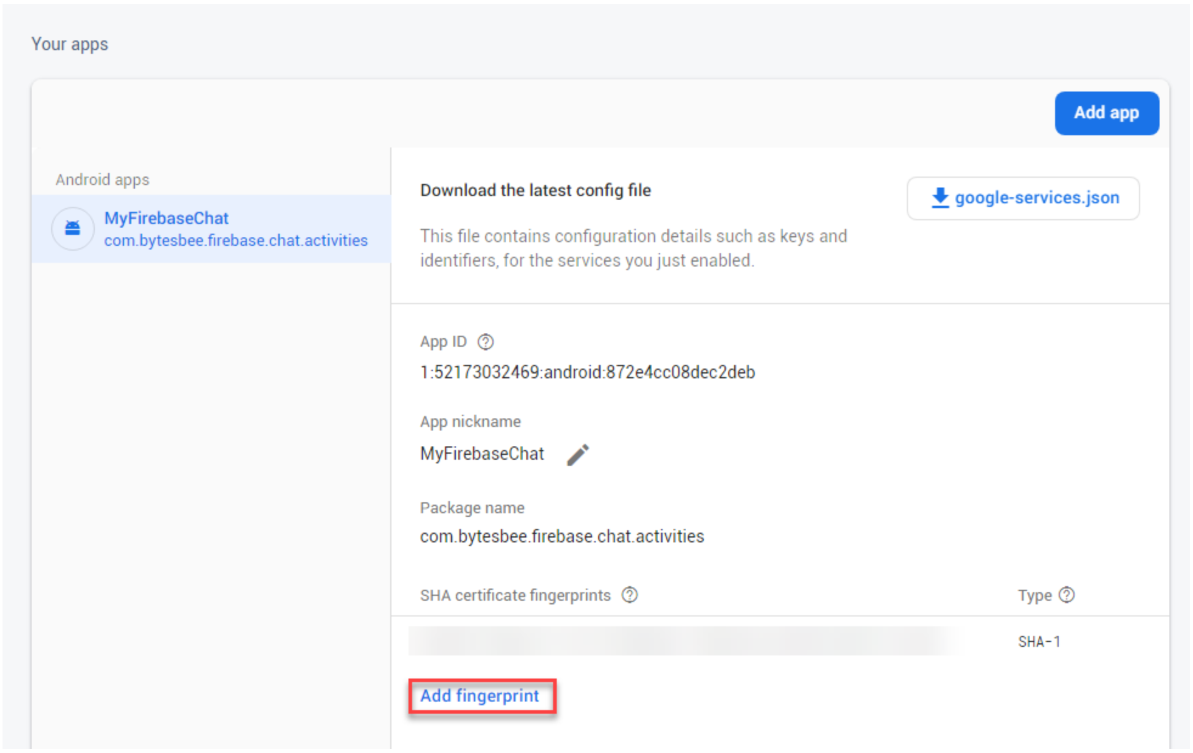


Guard your data with rules that define who has access to it and how it is structured

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if true;
    }
  }
}
```

Step 15: Add SHA-1 Key

Add Fingerprint SHA-1 key using below command :



Download the latest config file

This file contains configuration details such as keys and identifiers, for the services you just enabled.

App ID
1:52173032469:android:872e4cc08dec2deb

App nickname
MyFirebaseChat

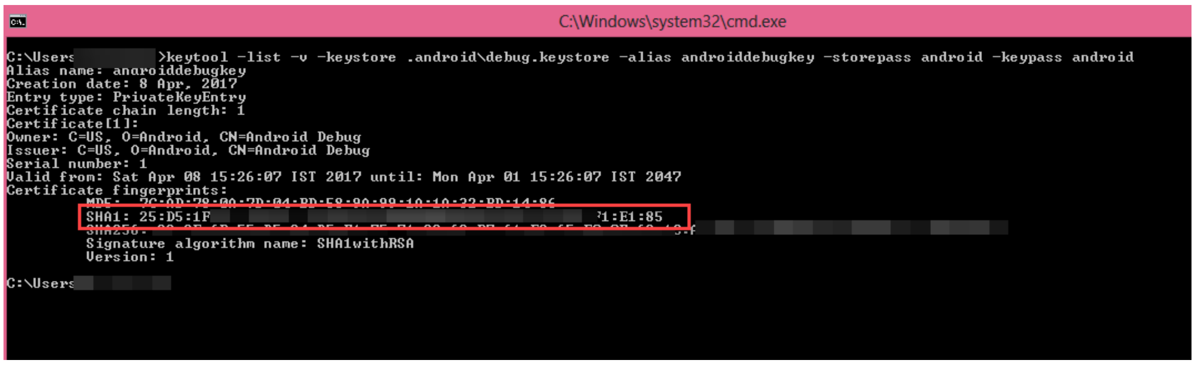
Package name
com.bytesbee.firebase.chat.activities

SHA certificate fingerprints | Type
[redacted] | SHA-1

Step 16: SHA-1 command

Add this command to get SHA certification (This command generating Debug key.)

```
keytool -list -v -keystore "C:\Users\{USER_NAME}\.android\debug.keystore" -alias
androiddebugkey -storepass android -keypass android
```



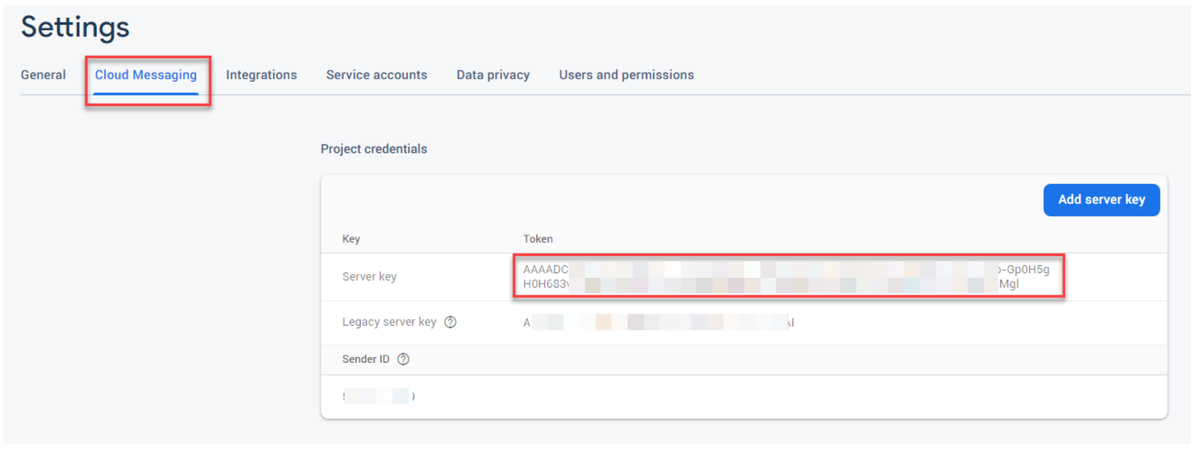
```
C:\Users[redacted]>keytool -list -v -keystore .android\debug.keystore -alias androiddebugkey -storepass android -keypass android
Alias name: androiddebugkey
Creation date: 8 Apr, 2017
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=US, O=Android, CN=Android Debug
Issuer: C=US, O=Android, CN=Android Debug
Serial number: 1
Valid from: Sat Apr 08 15:26:07 IST 2017 until: Mon Apr 01 15:26:07 IST 2047
Certificate fingerprints:
         MD5:  7C:AD:78:0A:7D:04:BD:58:90:99:10:1A:32:BD:14:86
         SHA1: 25:D5:1F[redacted]1:E1:85
         SHA256: [redacted]
         Signature algorithm name: SHA1withRSA
         Version: 1

C:\Users[redacted]
```

Note: Make sure to include the **RELEASE SHA-1** key to Firebase Console as well if you intend to create a RELEASE version for the Play Store. Otherwise, the creation of SHA-1 for the Release version is not required.

Important: Please download the most recent **google-service.json** and replace it in your Android project whenever a SHA-1 is added, updated, or removed from the Firebase Console for the best results.

Step 17: Cloud Messaging(CM) Key

Copy this Server key and paste into Android(Shown in next step)


Settings

General | Cloud Messaging | Integrations | Service accounts | Data privacy | Users and permissions

Project credentials

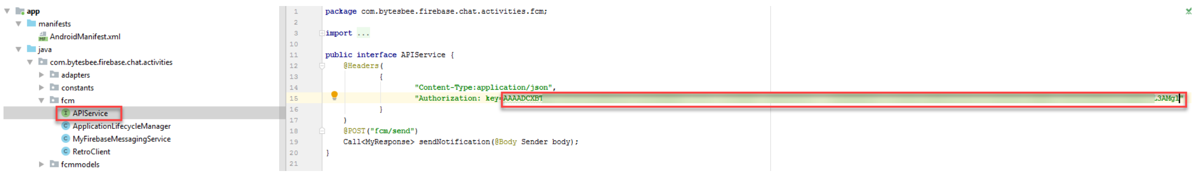
Add server key

Key	Token
Server key	AAAADC [redacted] -Gp0H5g H0HG83 [redacted] Mgl
Legacy server key	A [redacted] l
Sender ID	
[redacted]	

Step 18: Set CM key in Android

Paste above Server key into Android app -> APIService.java file here:

```
Authorization: key=PASTE_HERE
```

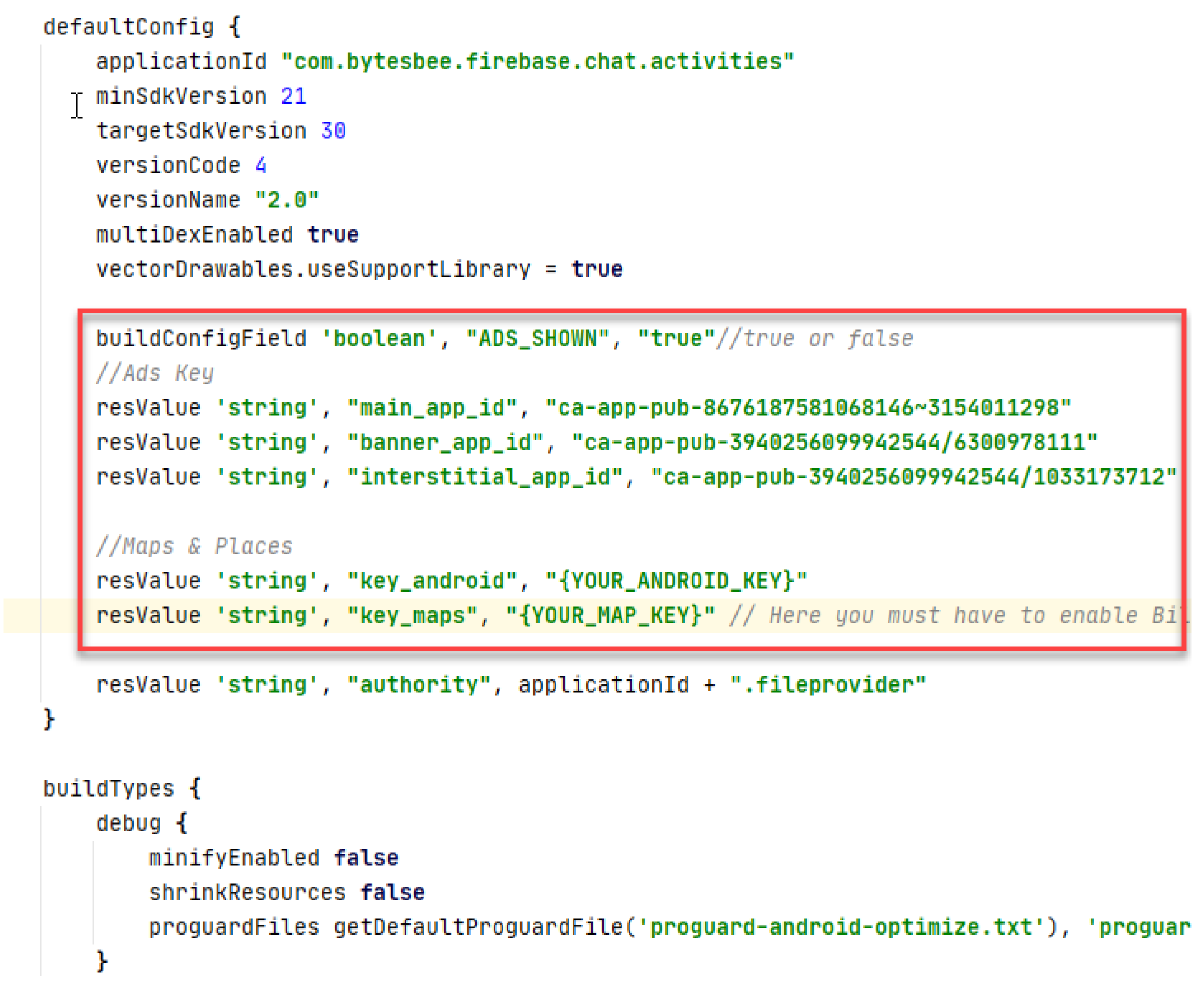


app
manifests
AndroidManifest.xml
java
com.bytesbee.firebase.chat.activities
adapters
constants
fcm
APIService
ApplicationLifecycleManager
MyFirebaseMessagingService
RetroClient
fcmmodels

```
package com.bytesbee.firebase.chat.activities.fcm;

import ...

public interface APIService {
    @Headers(
            {
                    "Content-Type:application/json",
                    "Authorization: key=AAAADCXBT [redacted] 3AMgl"
            }
    )
    @POST("fcm/send")
    Call<MyResponse> sendNotification(@Body Sender body);
}
```

Step 19: Configure Ads & Map

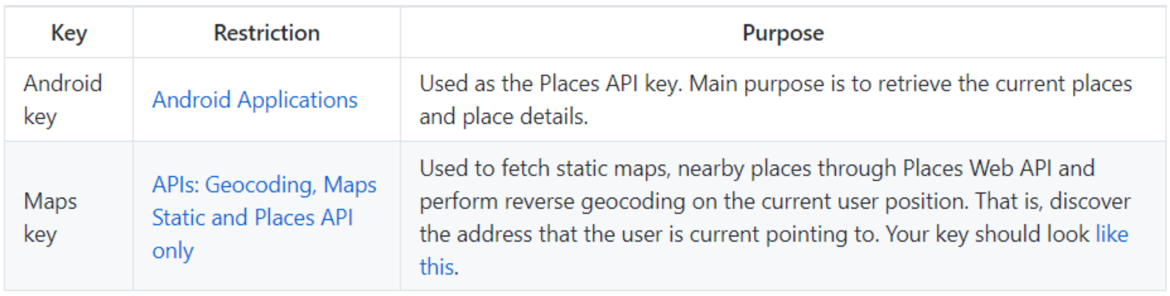
Change below lines for ADS and LOCATION related things



```
defaultConfig {
    applicationId "com.bytesbee.firebase.chat.activities"
    minSdkVersion 21
    targetSdkVersion 30
    versionCode 4
    versionName "2.0"
    multiDexEnabled true
    vectorDrawables.useSupportLibrary = true

    buildConfigField 'boolean', "ADS_SHOWN", "true"//true or false
    //Ads Key
    resValue 'string', "main_app_id", "ca-app-pub-8676187581068146~3154011298"
    resValue 'string', "banner_app_id", "ca-app-pub-3940256099942544/6300978111"
    resValue 'string', "interstitial_app_id", "ca-app-pub-3940256099942544/1033173712"

    //Maps & Places
    resValue 'string', "key_android", "{YOUR_ANDROID_KEY}"
    resValue 'string', "key_maps", "{YOUR_MAP_KEY}" // Here you must have to enable Bi

    resValue 'string', "authority", applicationId + ".fileprovider"
}

buildTypes {
    debug {
        minifyEnabled false
        shrinkResources false
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguar
    }
```

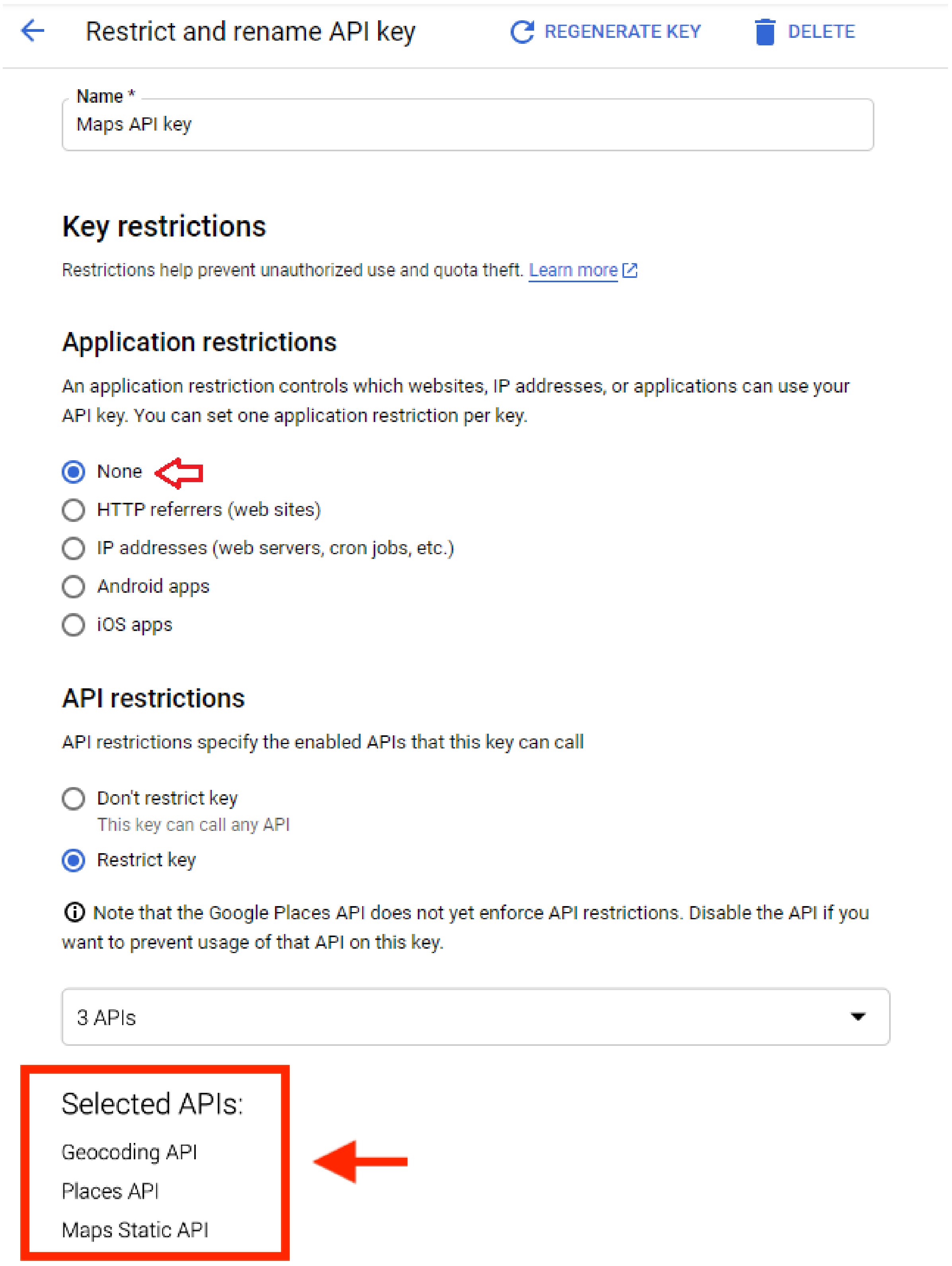
key_android: Enable Place API; [Click Here]

key_maps: Enable Geocoding API, Place API, Maps Static API (For MAPS KEY you can do below things and make sure you must have to Enable Billing first.) [Click Here]



Key	Restriction	Purpose
Android key	Android Applications	Used as the Places API key. Main purpose is to retrieve the current places and place details.
Maps key	APIs: Geocoding, Maps Static and Places API only	Used to fetch static maps, nearby places through Places Web API and perform reverse geocoding on the current user position. That is, discover the address that the user is current pointing to. Your key should look like this.

For MAPS KEY you can do below things and make sure you must have to **Enable Billing first.**


Restrict and rename API key

Name *
Maps API key

Key restrictions

Restrictions help prevent unauthorized use and quota theft. Learn more

Application restrictions

An application restriction controls which websites, IP addresses, or applications can use your API key. You can set one application restriction per key.

[✓ None]
HTTP referrers (web sites)
IP addresses (web servers, cron jobs, etc.)
Android apps
iOS apps

API restrictions

API restrictions specify the enabled APIs that this key can call

Don't restrict key
This key can call any API
[✓ Restrict key]

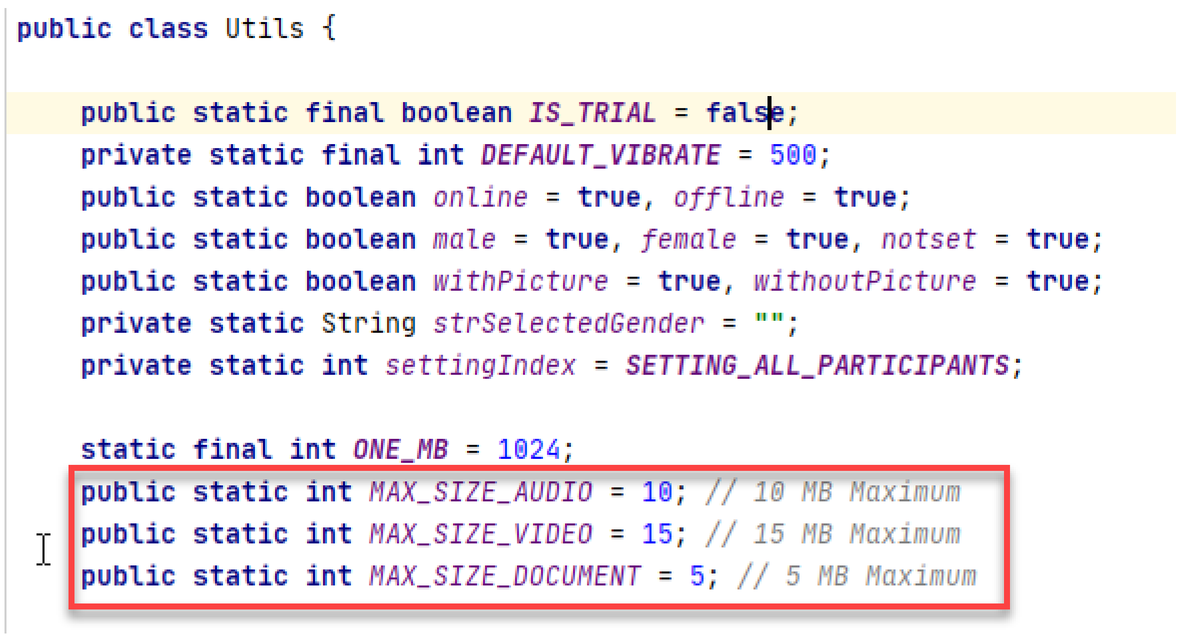
Note that the Google Places API does not yet enforce API restrictions. Disable the API if you want to prevent usage of that API on this key.

3 APIs

Selected APIs:
Geocoding API
Places API
Maps Static API

Step 20: Maximum Size

You can modify the MAXIMUM size of the documents, video and audio file from here.



```java
public class Utils {

    public static final boolean IS_TRIAL = false;
    private static final int DEFAULT_VIBRATE = 500;
    public static boolean online = true, offline = true;
    public static boolean male = true, female = true, notset = true;
    public static boolean withPicture = true, withoutPicture = true;
    private static String strSelectedGender = "";
    private static int settingIndex = SETTING_ALL_PARTICIPANTS;

    static final int ONE_MB = 1024;
    public static int MAX_SIZE_AUDIO = 10; // 10 MB Maximum
    public static int MAX_SIZE_VIDEO = 15; // 15 MB Maximum
    public static int MAX_SIZE_DOCUMENT = 5; // 5 MB Maximum
```

Step 21: Done

That's it. All Done. Congrats! Now you can run your project with new Firebase setup in your server.

Thanks for purchasing. Hope you like it! If you like it, please don't forget to give your valuable feedback and star rating for this script.

Have a great day!
BytesBee